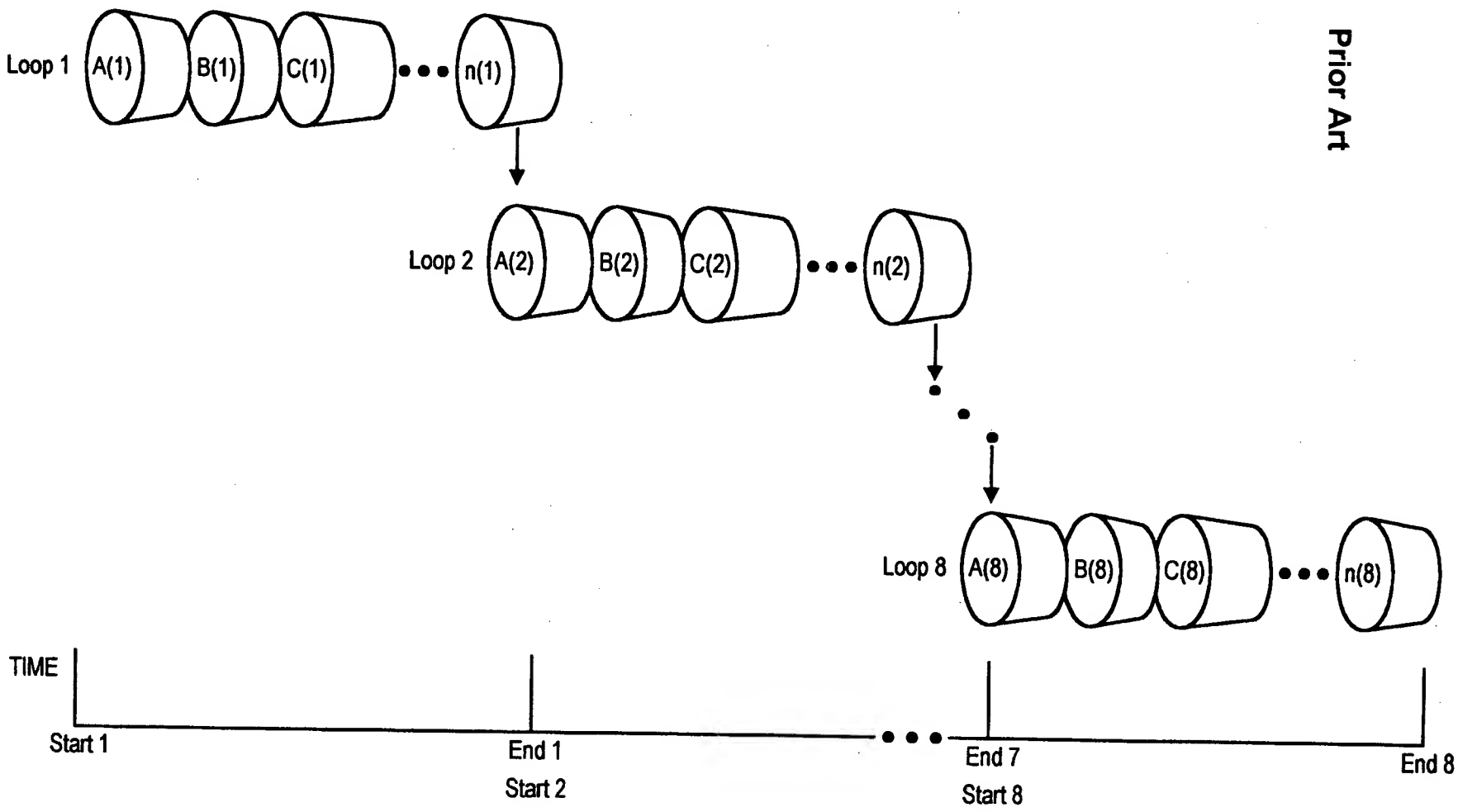


Prior Art

Figure 1

Prior Art

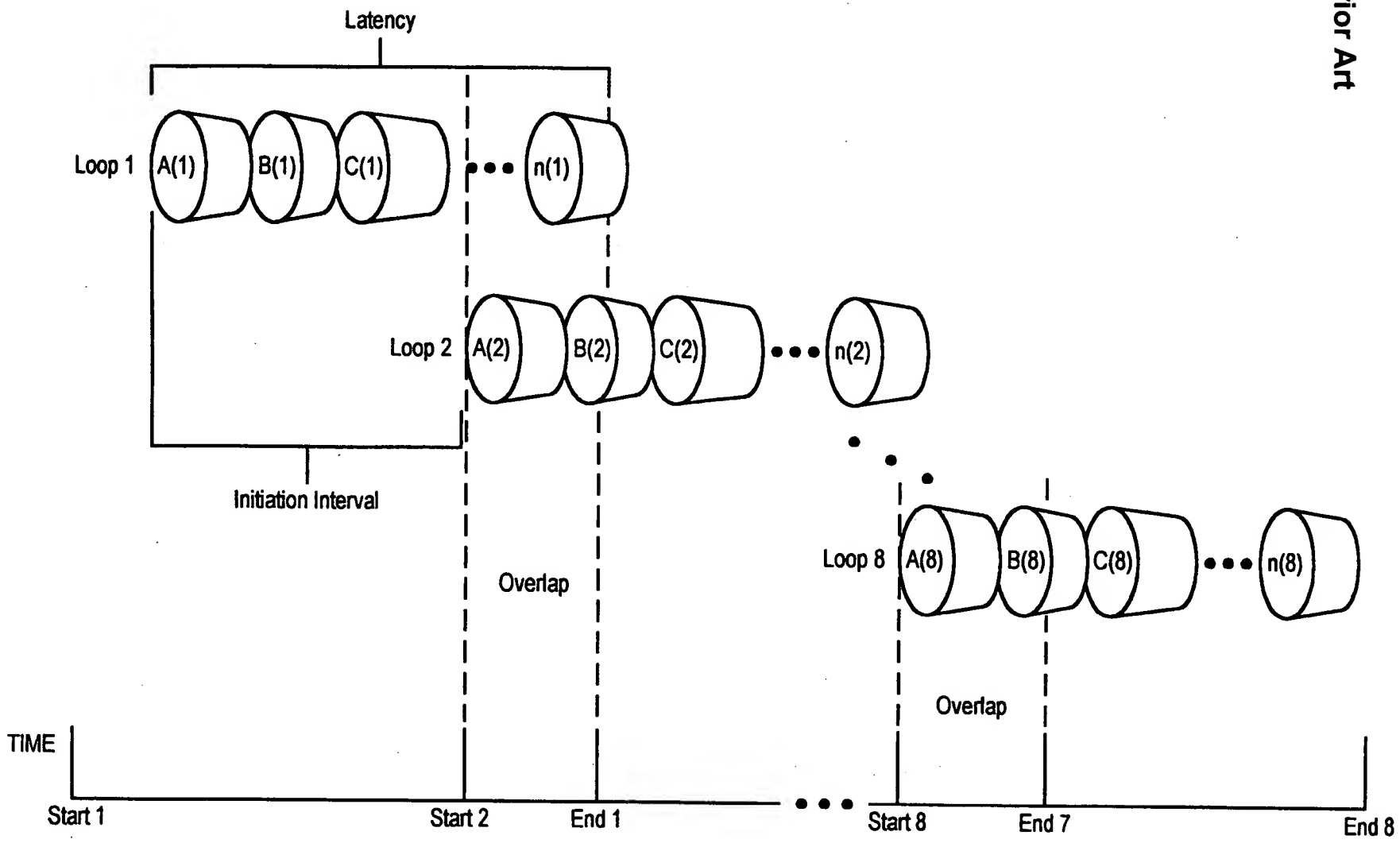


Figure 2

Prior Art

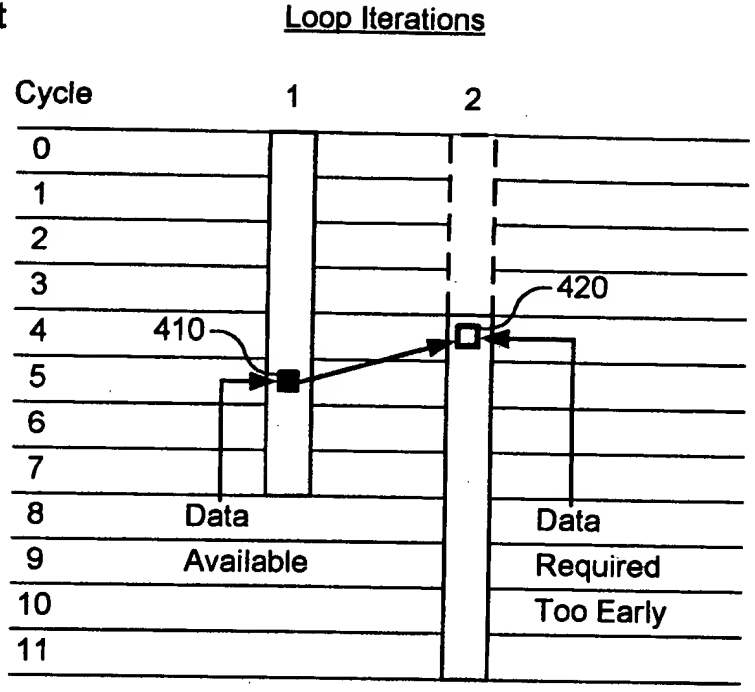


Figure 3 (a)

Prior Art

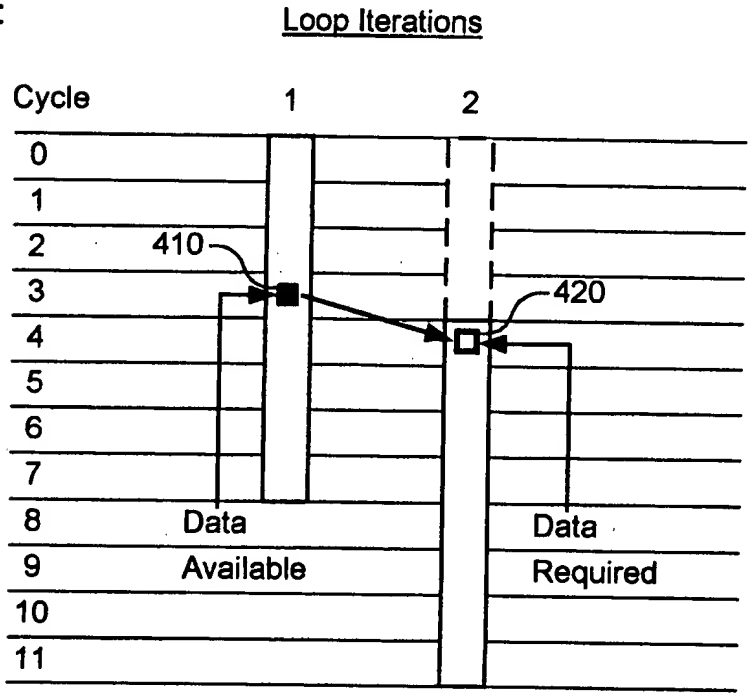


Figure 3 (b)

Prior Art

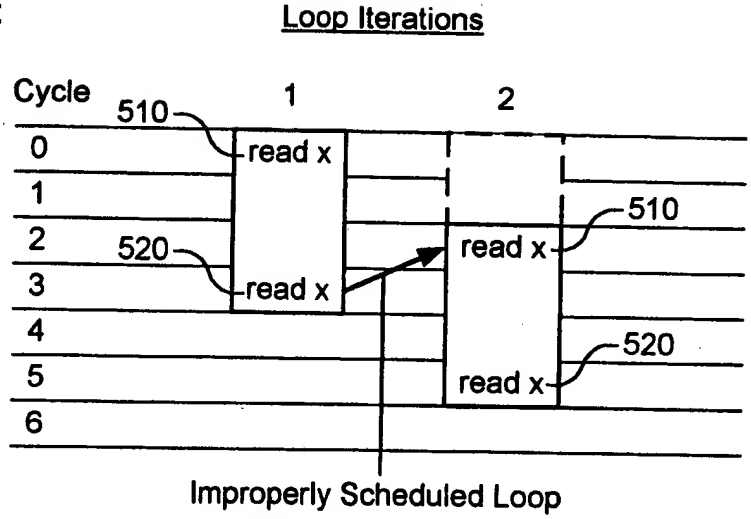


Figure 4 (a)

Prior Art

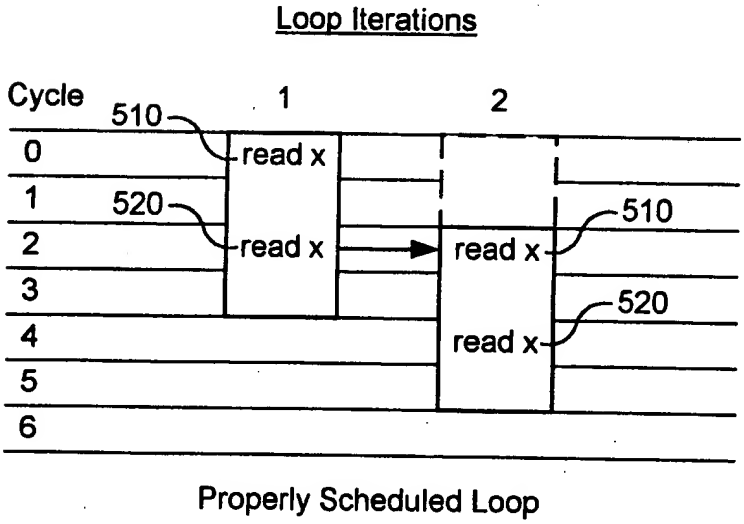


Figure 4 (b)

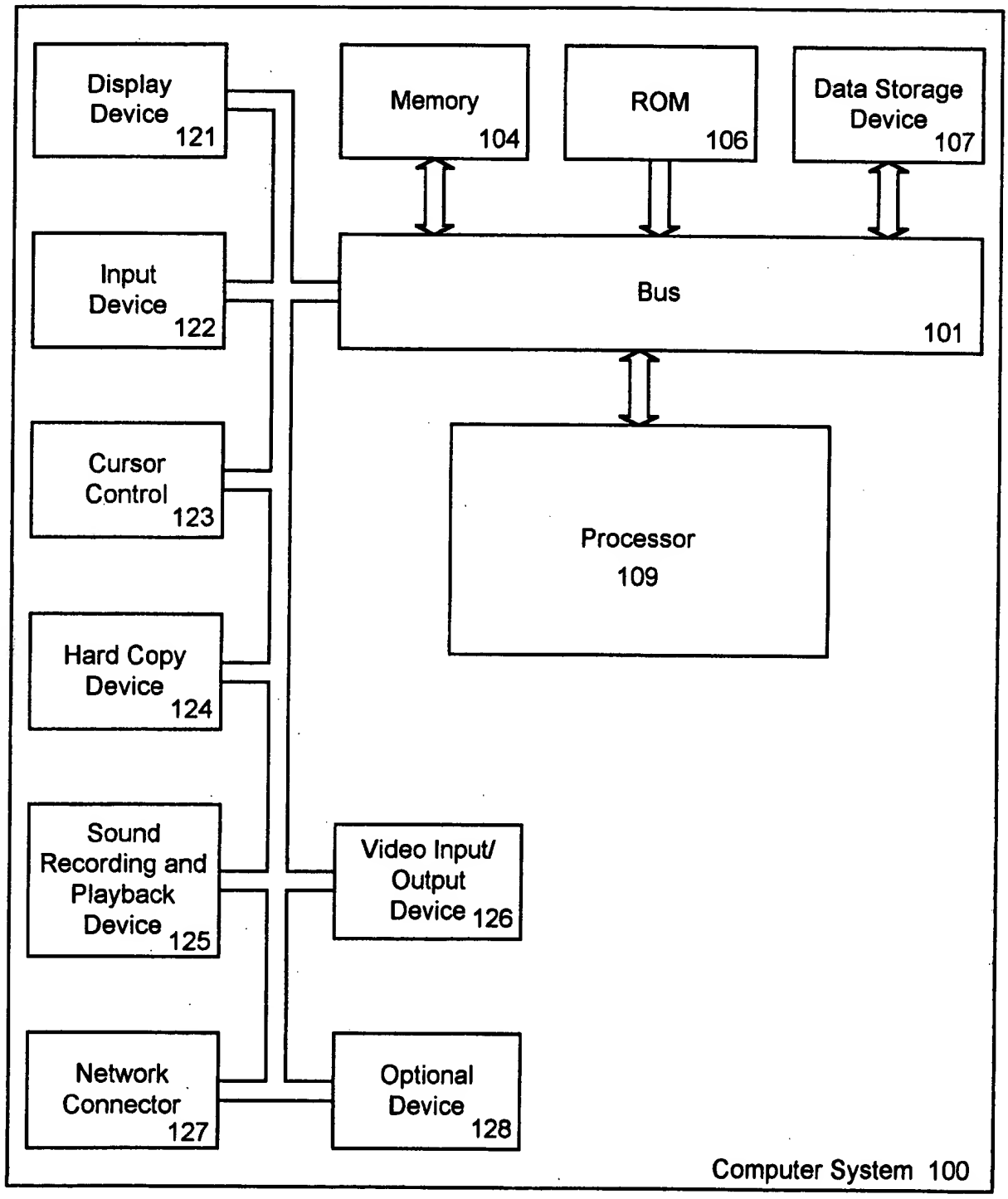


Figure 5

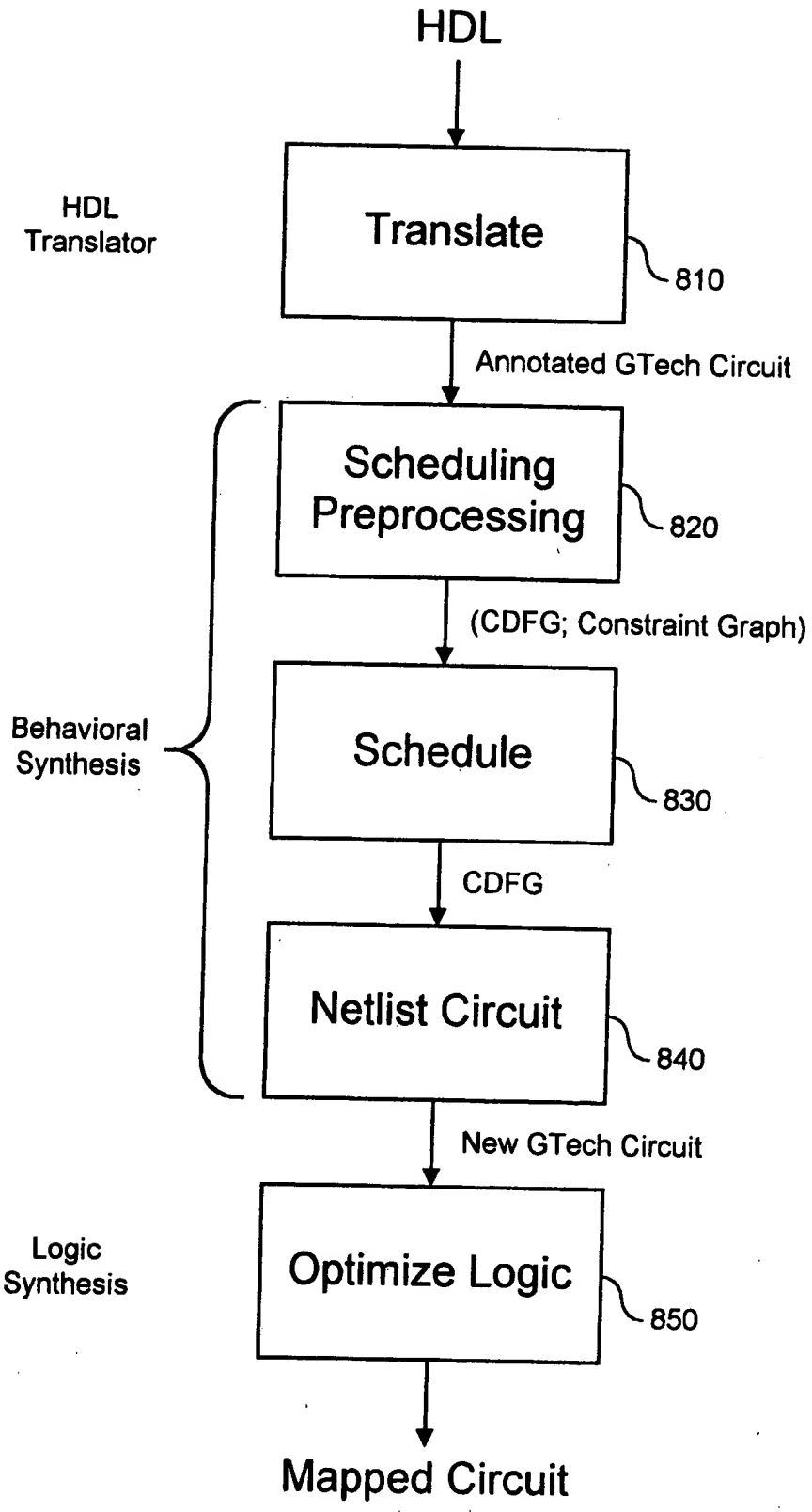


Figure 6 Synthesis with Scheduling

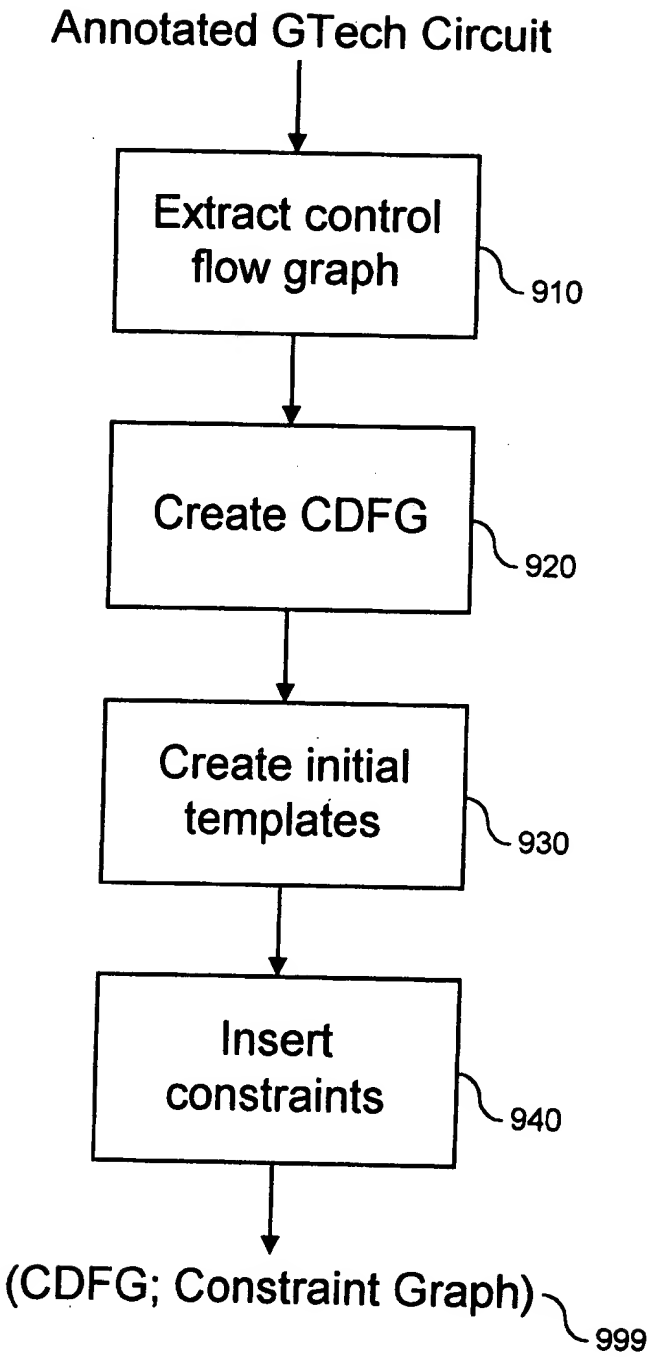


Figure 7 Scheduling Preprocessing

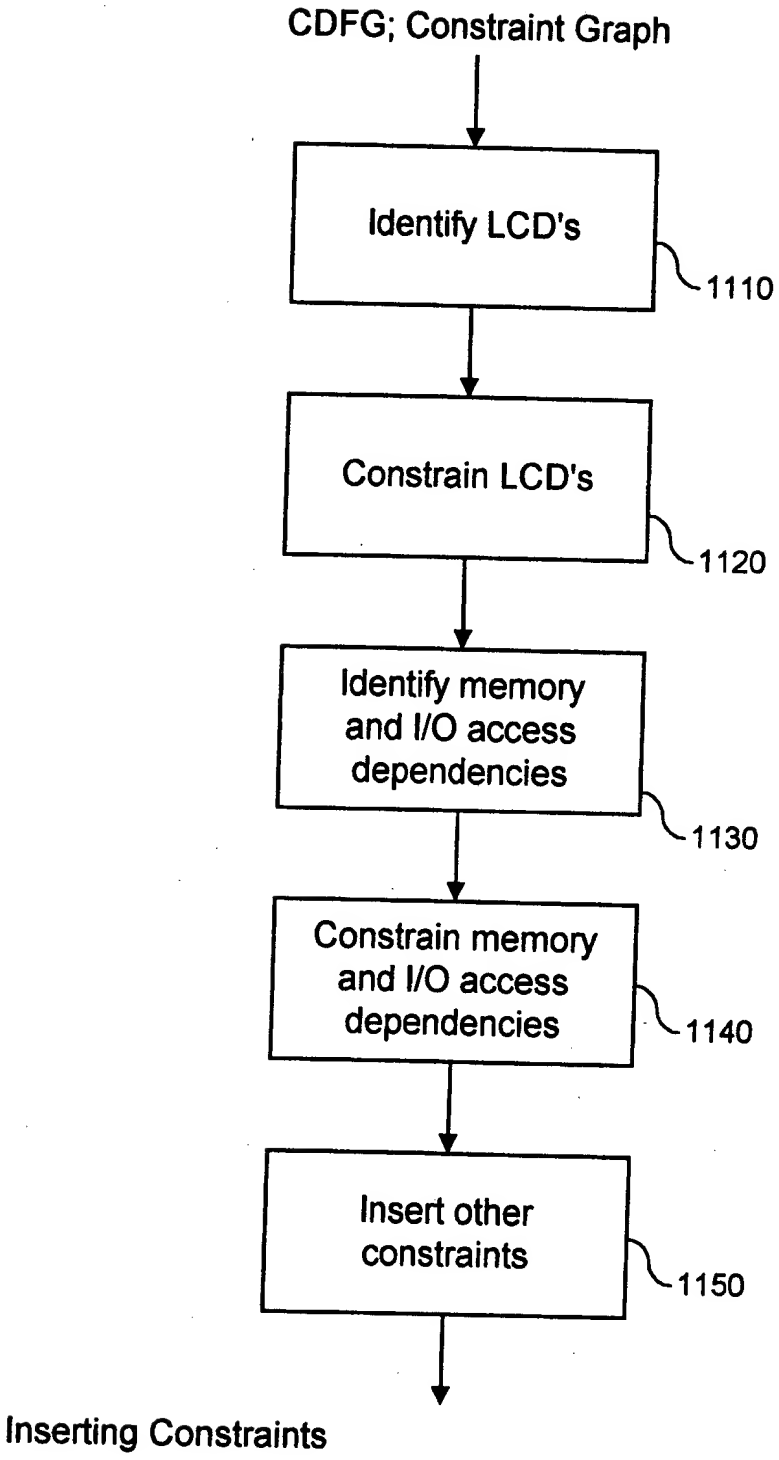
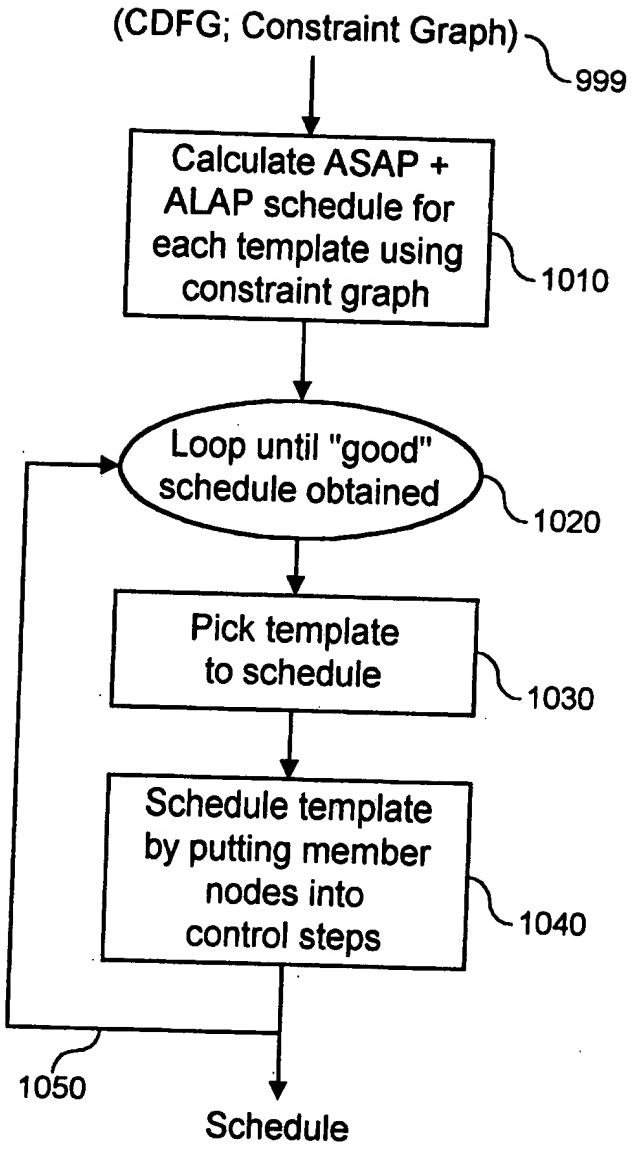


Figure 8



Scheduling Using Templates

Figure 9

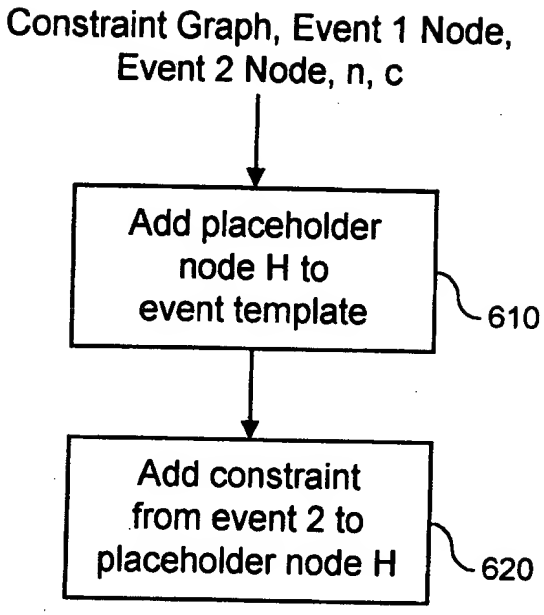


Figure 10

```
module loopex8 ( c, x, y, z, clock);
input [1:0] x, y, z;
    input clock ;
    output [2:0] c;
    reg [2:0] c;
    reg [2:0] p;

    always begin                                    3030

        forever begin : theloop                     3010

            c <= x - p ;

            @(posedge clock) ;                      3020

            p = y + z ;

            @(posedge clock) ;

        end

    end

endmodule
```

Figure 11

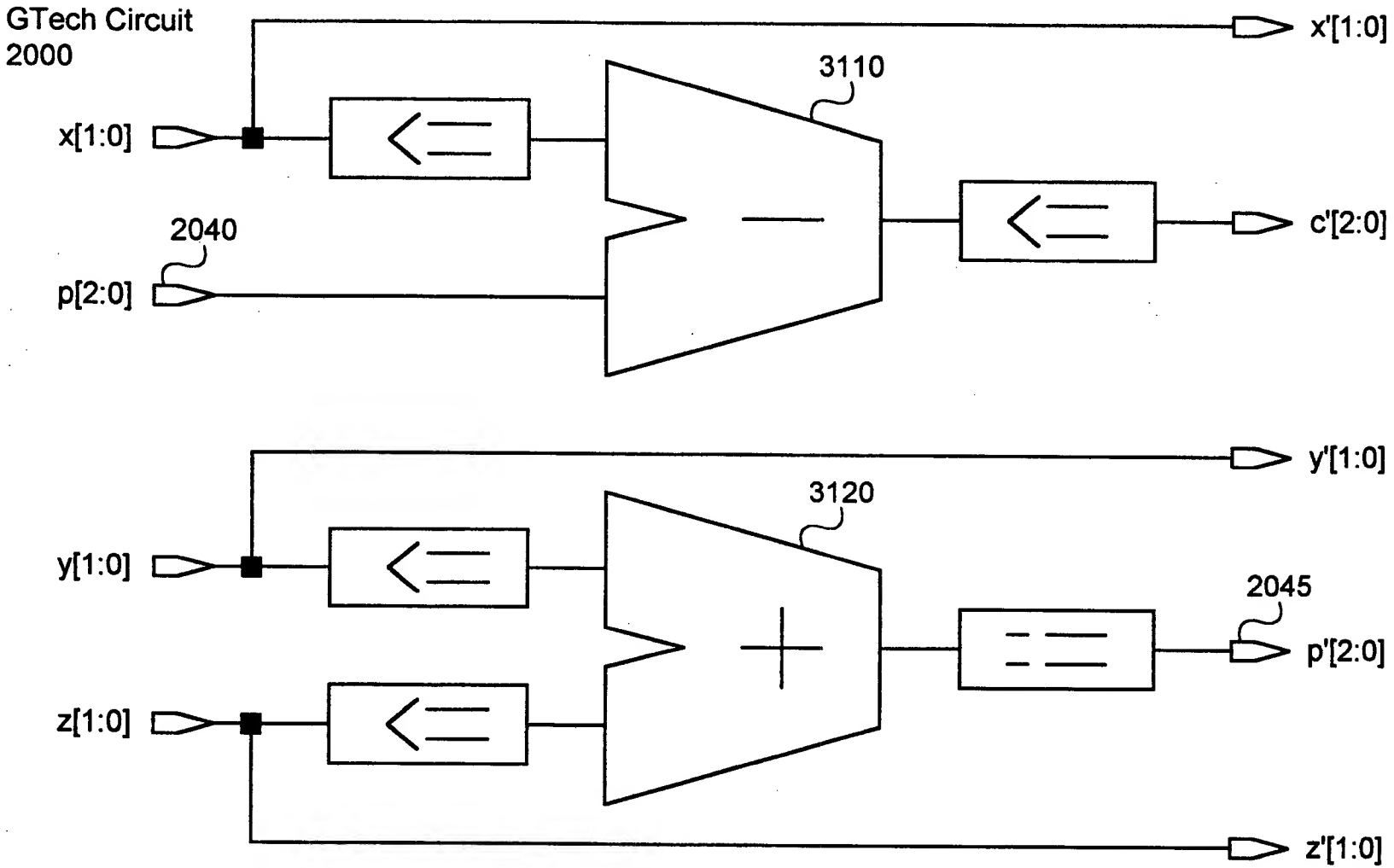
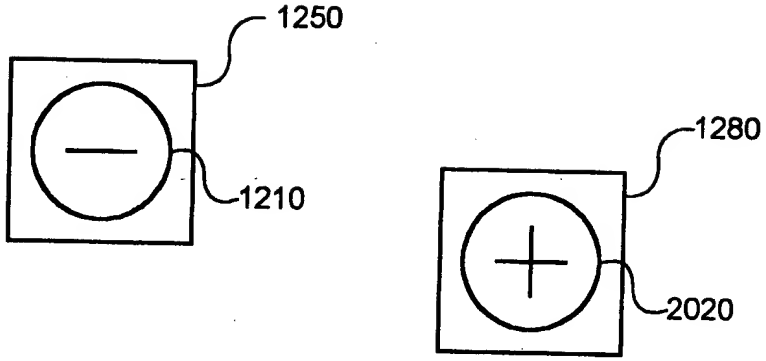


Figure 12



n = 2

Figure 13a

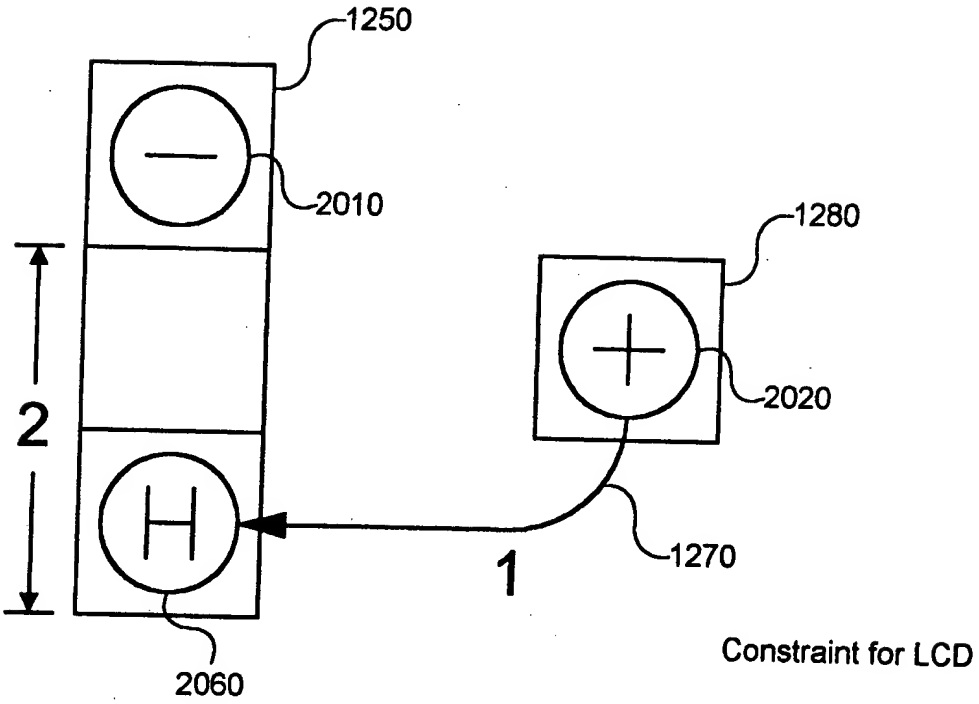


Figure 13b

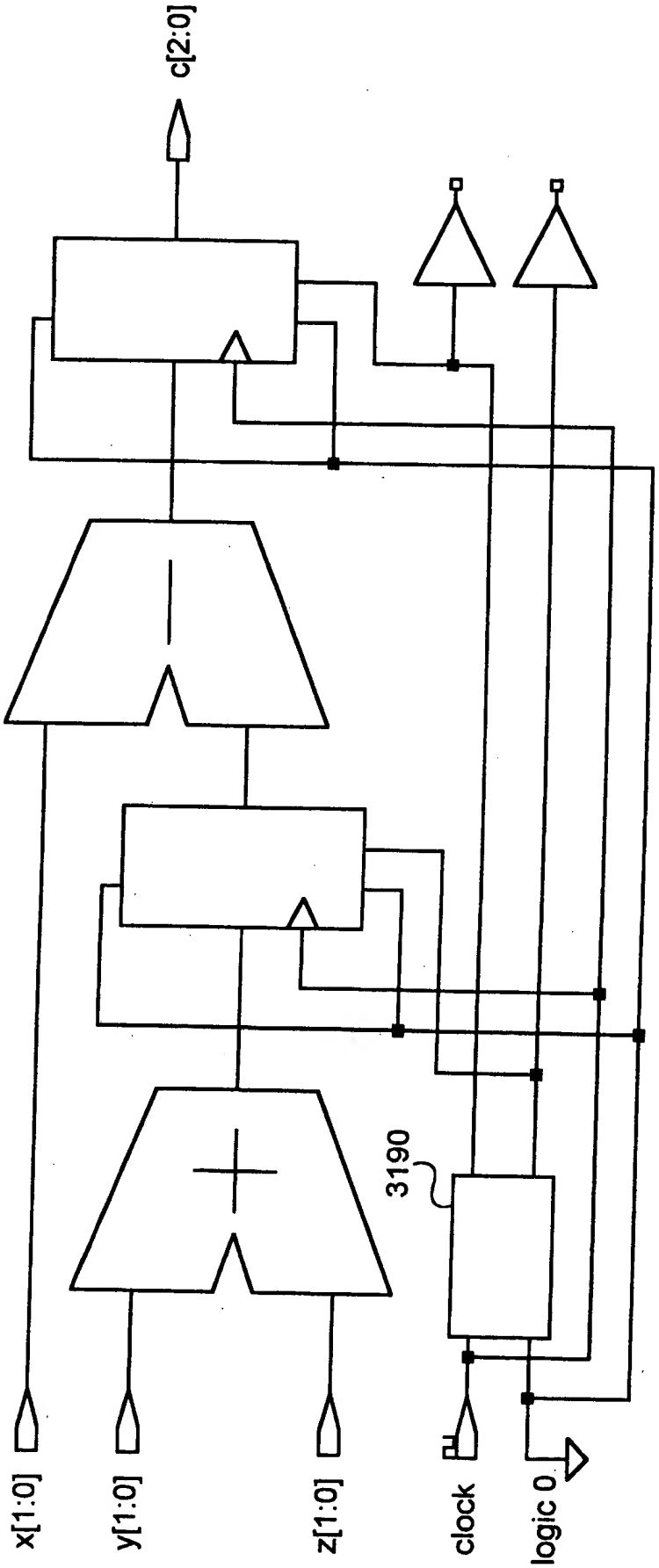


Figure 14

```
module write4 ( w, x, clock);

    input [15:0] x ;
    input clock ;
    output [31:0] w;
    reg [32:0] w;
    reg [15:0] x1;
    reg [15:0] x2;

    always begin                                    1530

        forever begin : writeloop                   1530

            x1 <= x ;

            @(posedge clock) ;                      1530

            x2 <= x ;

            w <= x1 * x2 ;

        end

    end

endmodule
```

Figure 15

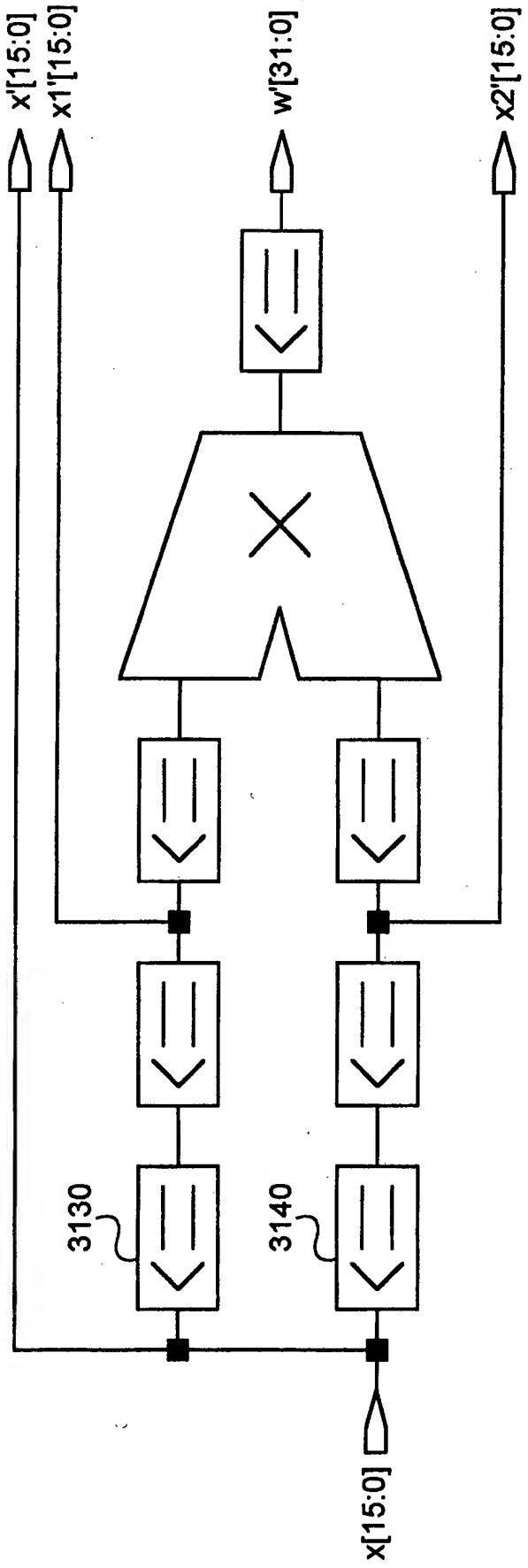


Figure 16

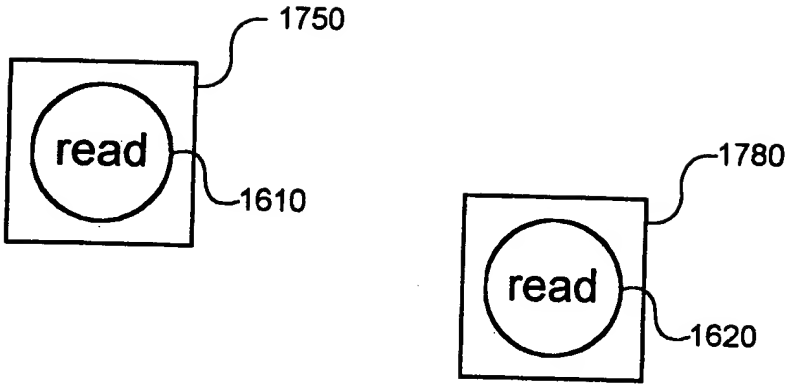


Figure 17a

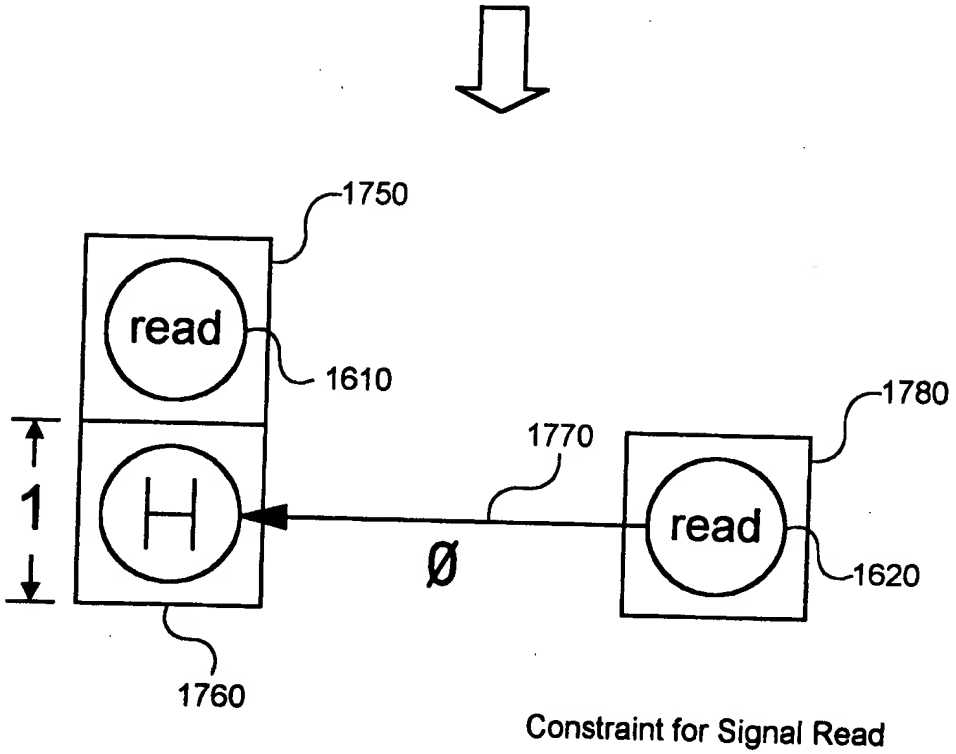


Figure 17b

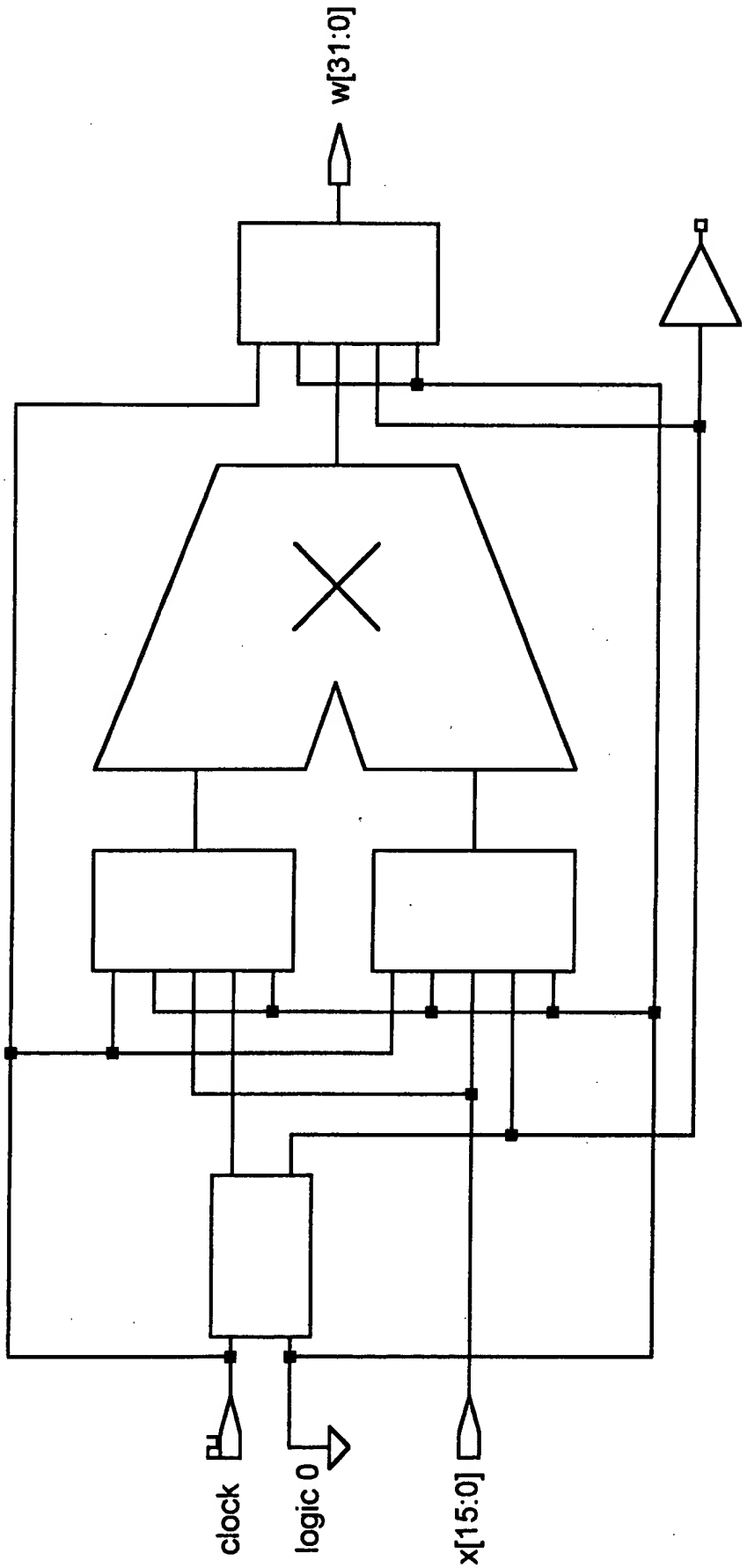


Figure 18

```
module after1 ( c, x, y, z, clock);

    input [1:0] x, y, z;
    input clock ;
    output [2:0] c;
    reg [2:0] c;
    reg [2:0] p;

    always begin

            @(posedge clock) ;

            forever begin
                c <= #24 x - p ;

                @(posedge clock) ;

                p = y + z ;

                @(posedge clock) ;
            end
    end
endmodule
```

Figure 19 (a)

```
entity after1 is
    port(
            c : out integer range 0 to 7;
            x, y, z : in integer range 0 to 3;
            clock : in bit
    );
end after1;

architecture behavioral of after1 is begin
    process
            variable p : integer range 0 to 7;
    begin
            wait until clock'event and clock = '1';

            loop

                c <= transport x - p after 24 ns;

                wait until clock'event and clock = '1';

                p := y + z;

                wait until clock'event and clock = '1';

            end loop;

   end process;
end behavioral;
```

Figure 19 (b)

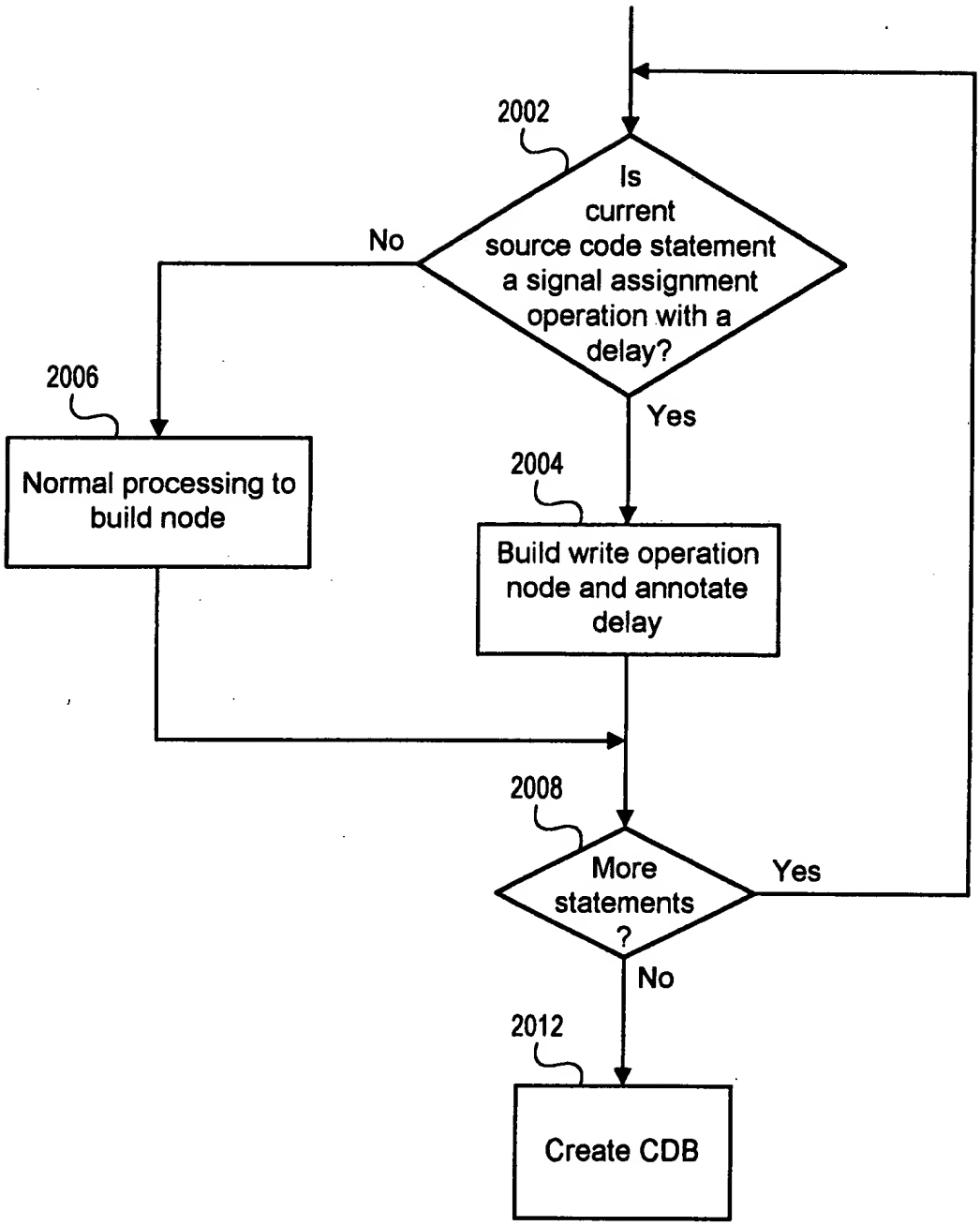


Figure 20

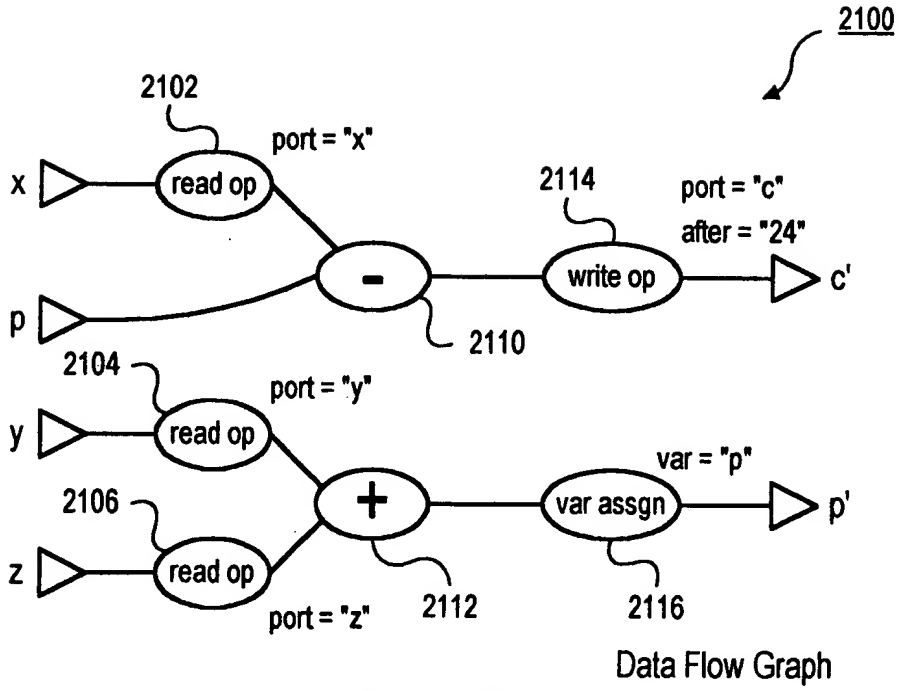


Figure 21

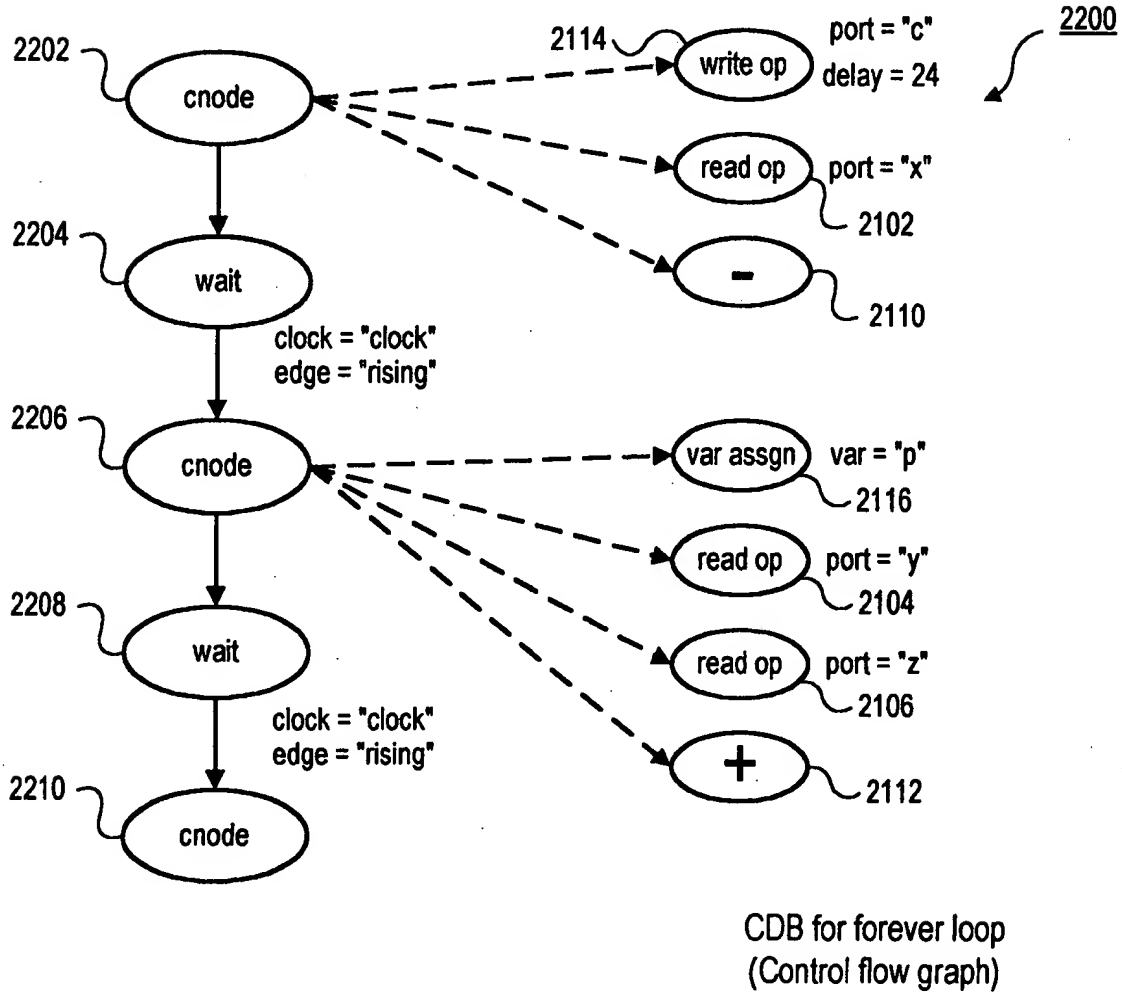


Figure 22

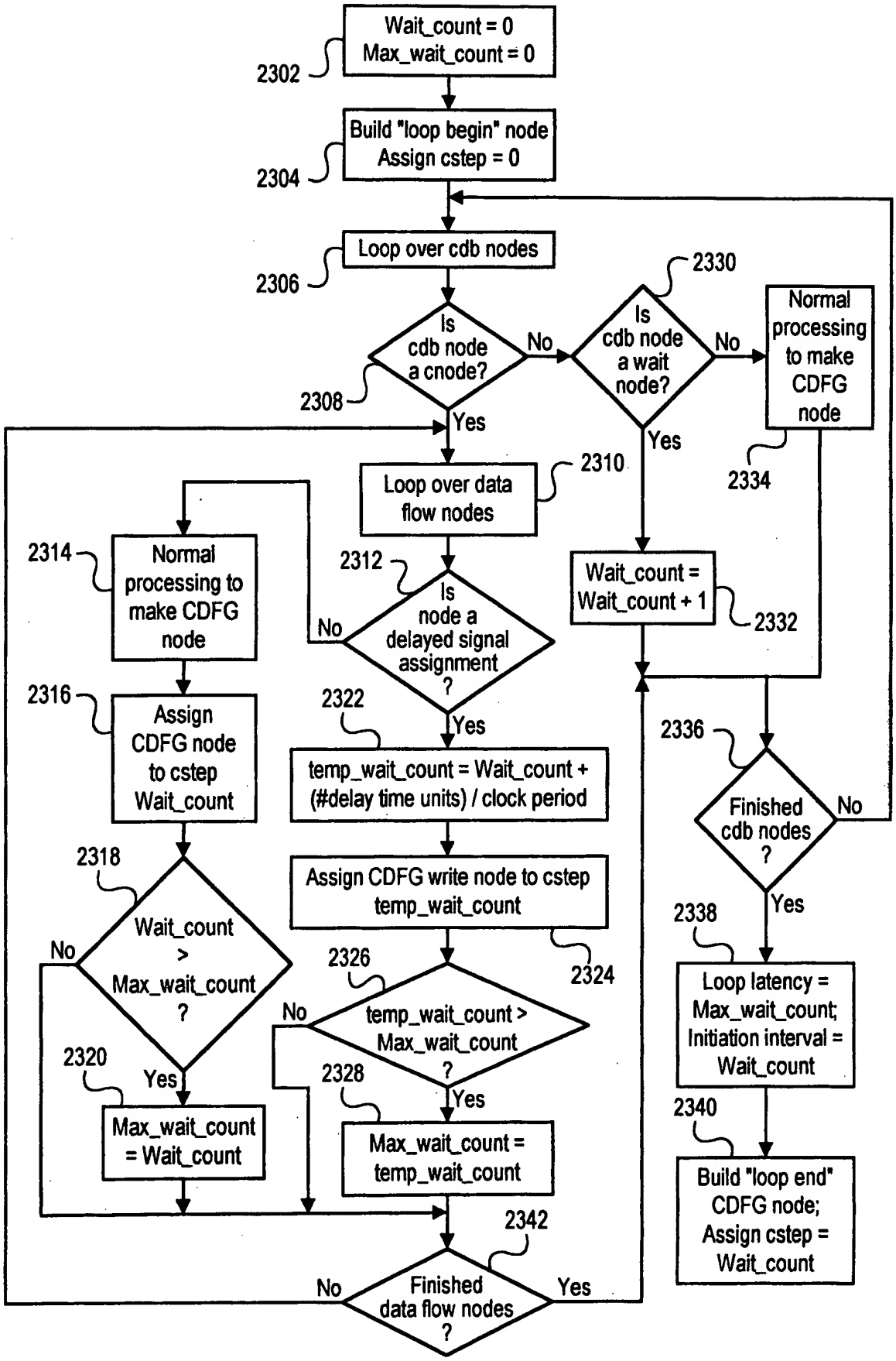


Figure 23

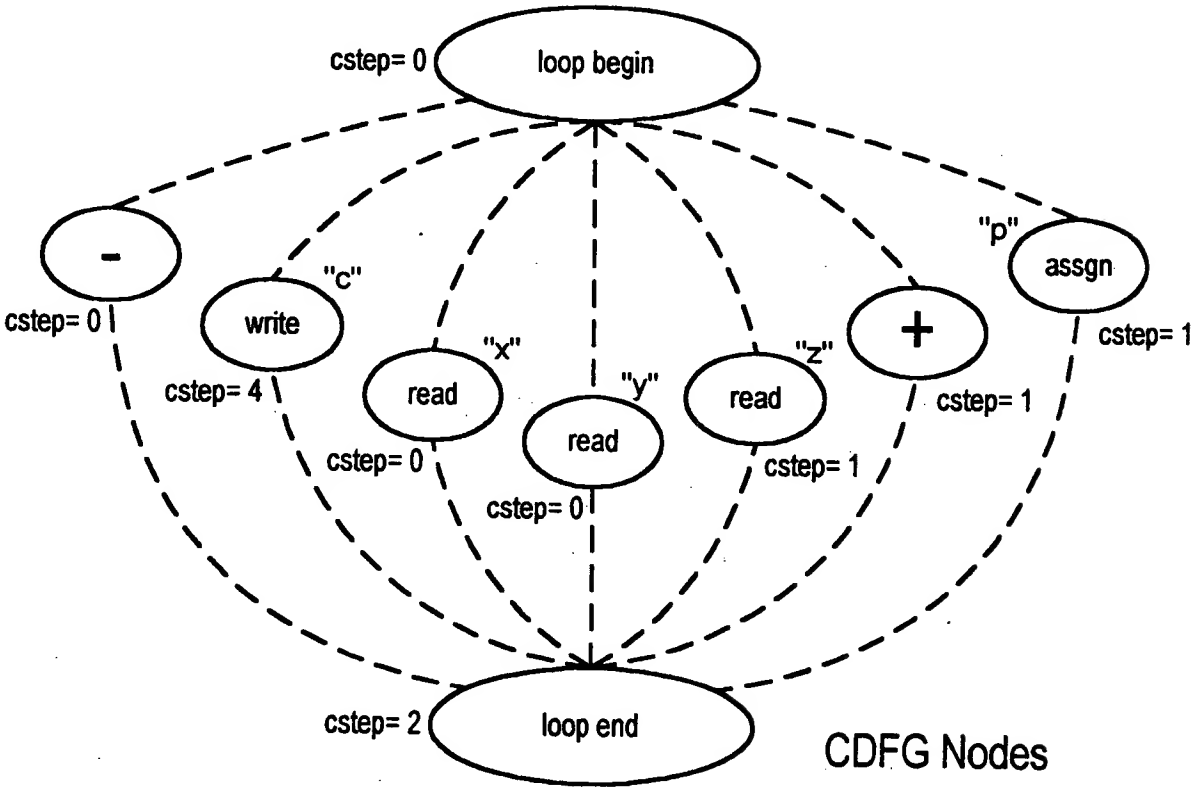


Figure 24

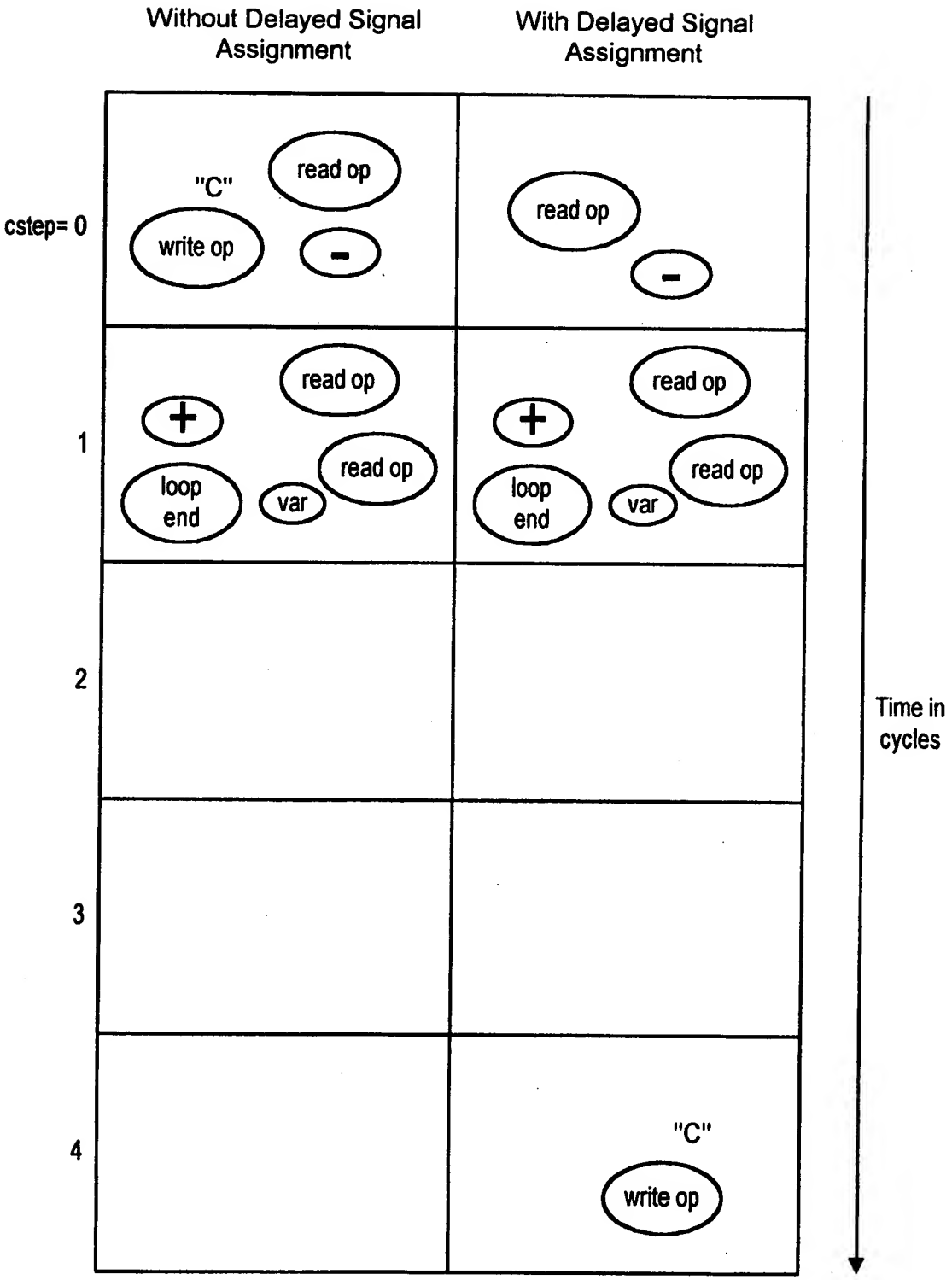


Figure 25

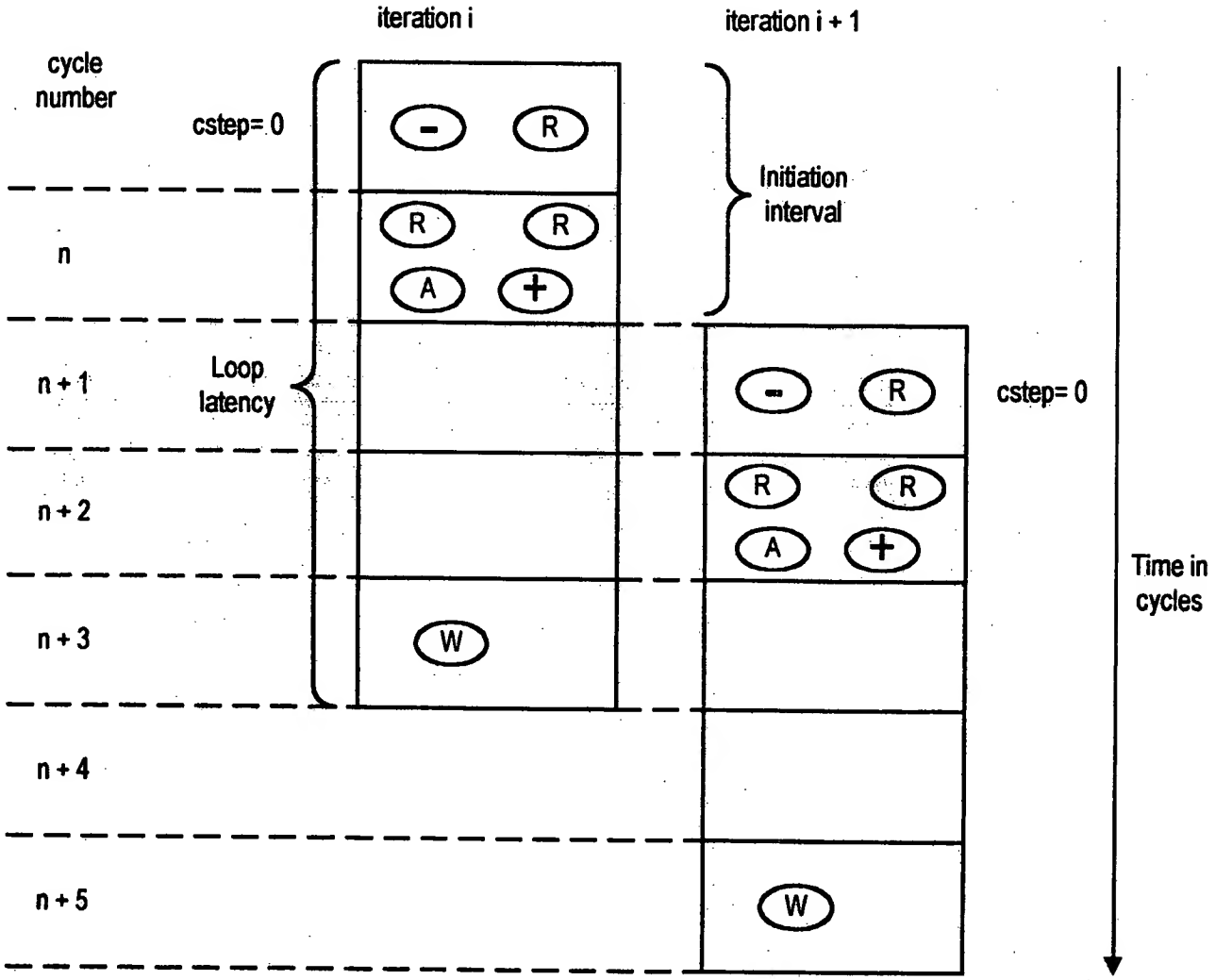


Figure 26

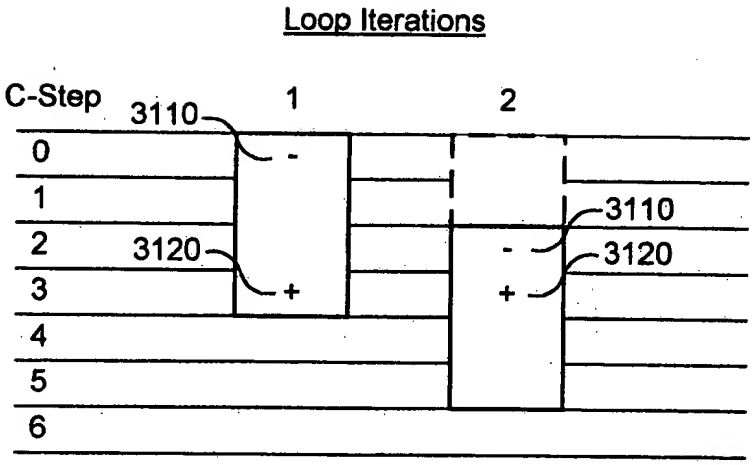


Figure 27

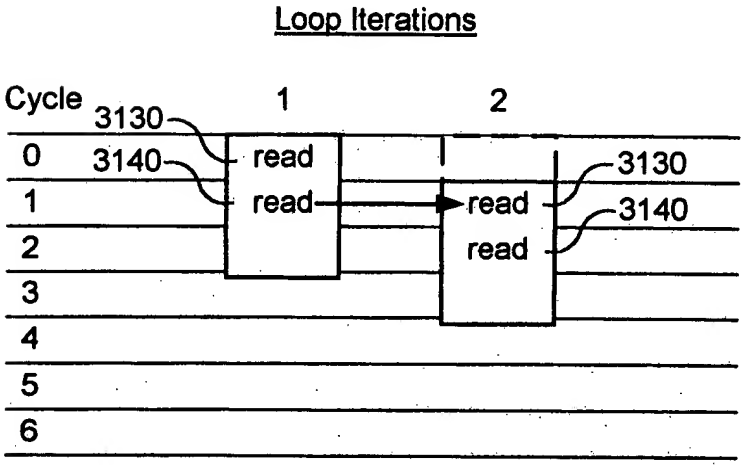
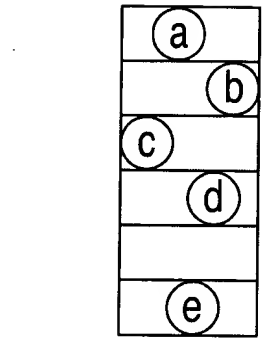


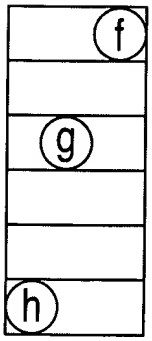
Figure 28



cdfg behavioral template

cdfg node

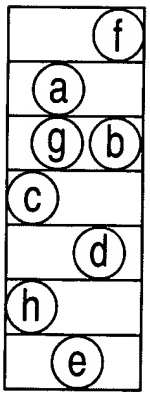
FIG. 29a



cdfg behavioral template

cdfg node

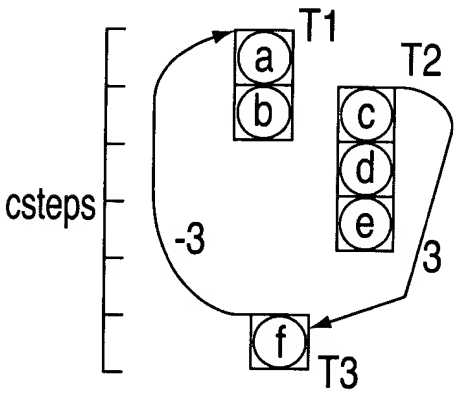
FIG. 29b



cdfg behavioral template

cdfg node

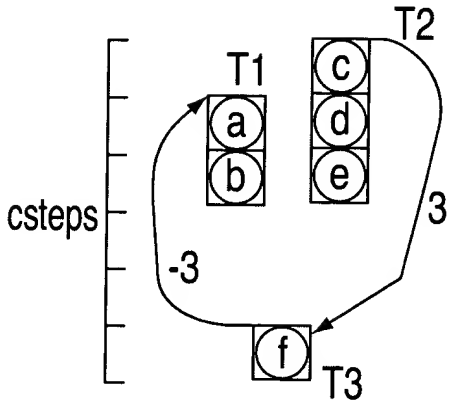
FIG. 29c



a uses same c resource

LP(T2,T1) = 0

FIG. 30a



a uses same c resource

LP(T2,T1) = 0

FIG. 30b

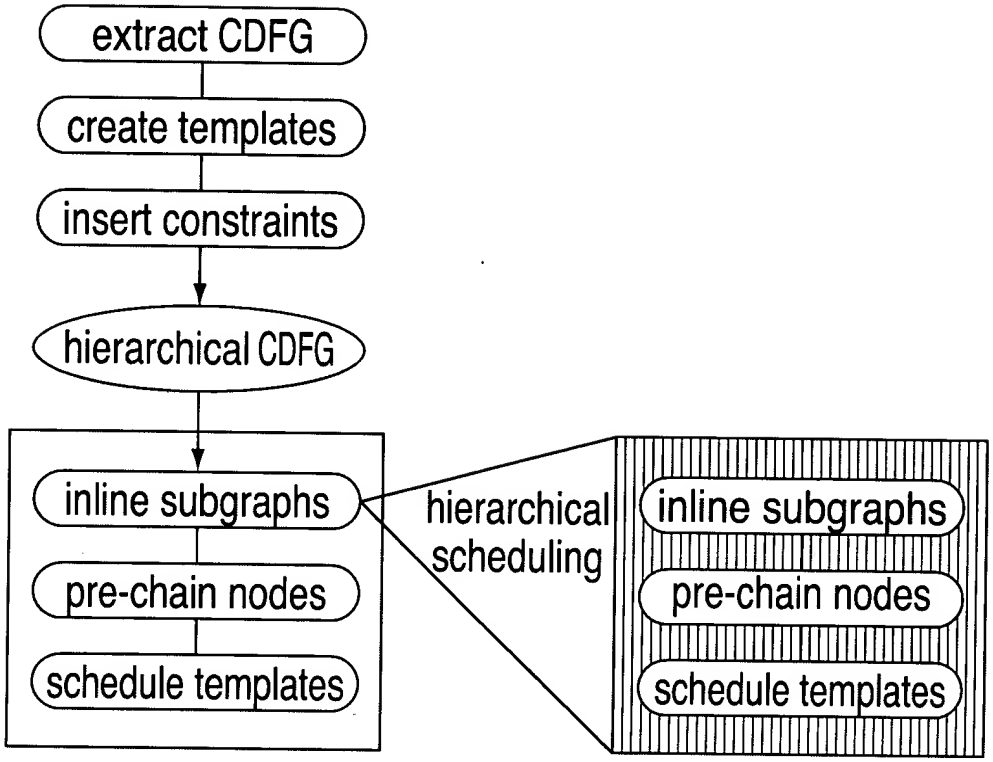


FIG. 31

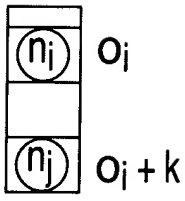


FIG. 32a

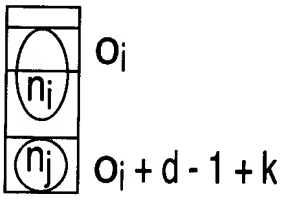


FIG. 32b

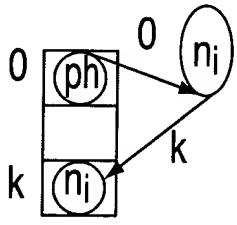


FIG. 32c

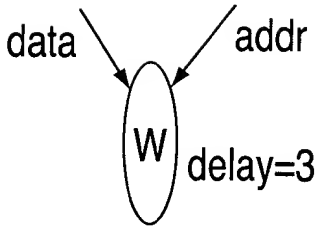


FIG. 33a

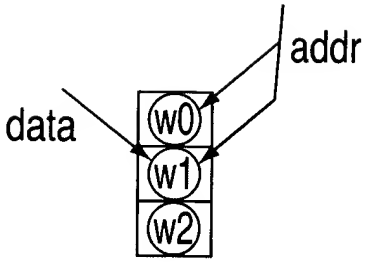


FIG. 33b

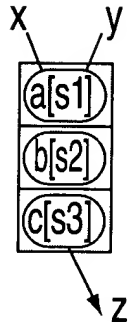


FIG. 34a

FIG. 34b

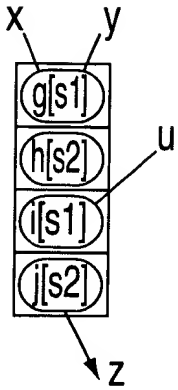


FIG. 34c

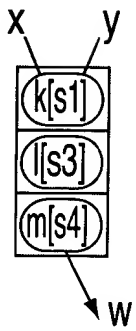


FIG. 34d

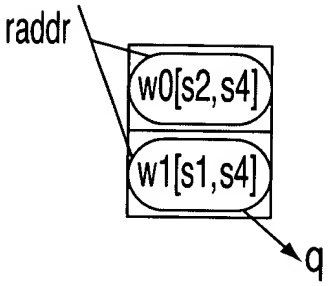


FIG. 35a

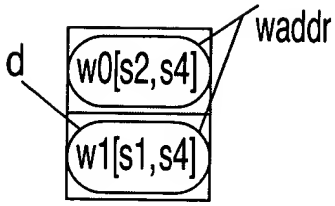
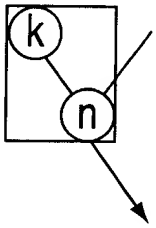
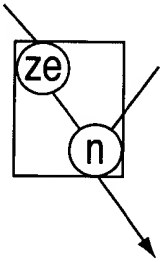


FIG. 35b



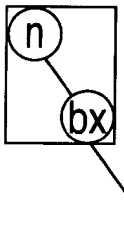
(k) constant
(ze) zero-extend
(bx) bit-extract
(u1) (u2) logic operations

FIG. 36a



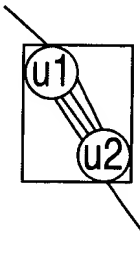
(k) constant
(ze) zero-extend
(bx) bit-extract
(u1) (u2) logic operations

FIG. 36b



(k) constant
(ze) zero-extend
(bx) bit-extract
(u1) (u2) logic operations

FIG. 36c



(k) constant
(ze) zero-extend
(bx) bit-extract
(u1) (u2) logic operations

FIG. 36d

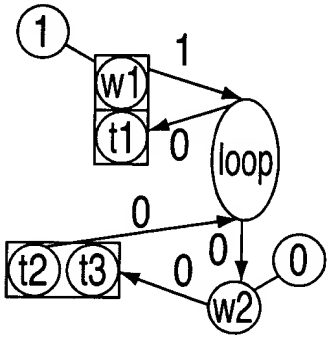


FIG. 37a

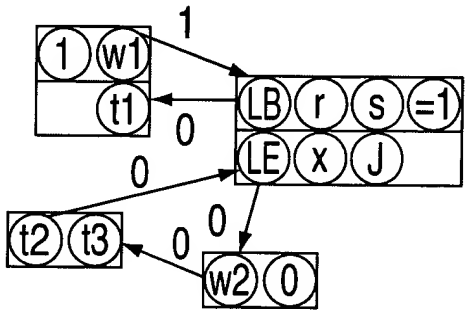


FIG. 37b

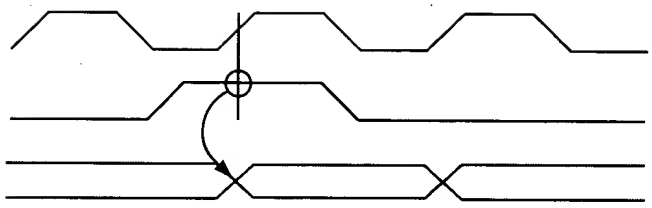


FIG. 38

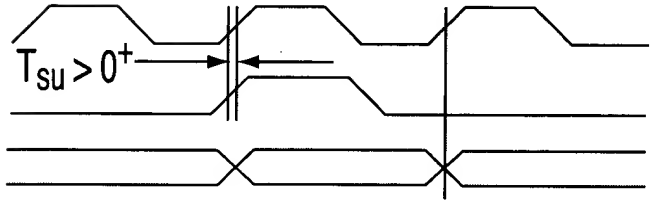


FIG. 39a

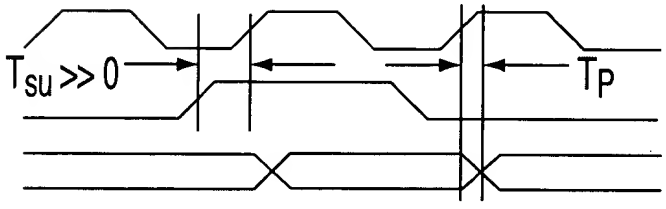


FIG. 39b

```
o1 <= v1;
while (c) begin: loop
    o2 <= v2;
    @(posedge clock);
end
o3 <= v3;
@(posedge clock);
```

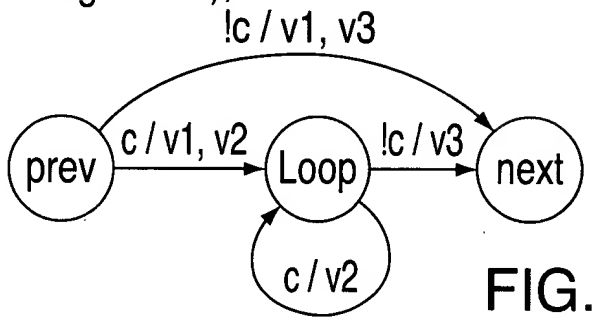


FIG. 40

```
o1 <= v1;
while (c) begin: loop
    @(posedge clock);
    o2 <= v2;
end
@(posedge clock);
o3 <= v3;
```

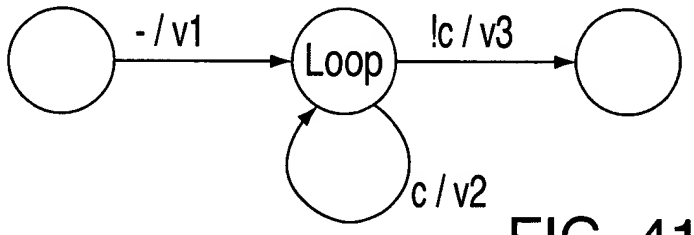


FIG. 41

```
@(posedge clock);
if (input_signal= 1'b1)begin
        x= input_read_1;
        y=input _read_2;
        tmp = x+ y;//2 cycle addition
        @(posedge clock);//strobe stab regs
        @(posedge clock);//1st cycle of add
        @(posedge clock);//2nd cycle of add
        out<=tmp;
end
@(posedge clock);
```

FIG. 42

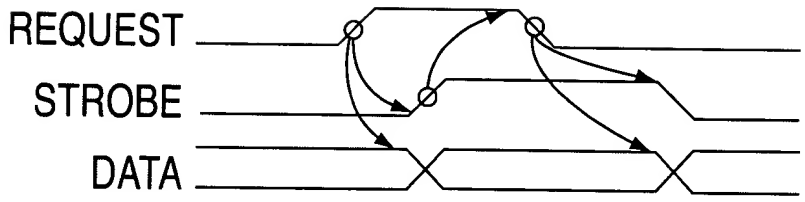


FIG. 43

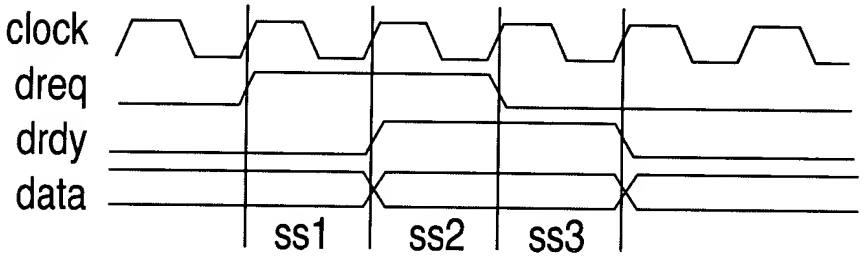


FIG. 44a

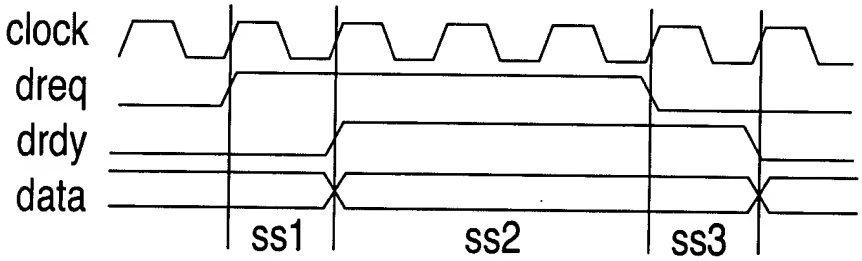


FIG. 44b

```
a1=in_port1;
a2=in_port2;
@(posedge clock);
out_port_1<=long_function_1(a1,a2);
@(posedge clock);
b1=in_port3;
b2=in_port4;
@(posedge clock);
out_port_2<=long_function_2(b1,b2);
```

FIG. 45